

(11)Publication number : 09-022361
(43)Date of publication of application : 21.01.1997

G06F 9/45

(72)Inventor : NISHIYAMA HIROYASU
KIKUCHI SUMIO

[illegible]

<http://www19.ipdl.ncipi.go.jp/PA1/result/detail/main/wAAAkWayZ3DA409022361...> 2005/12/22

THIS PAGE BLANK (USPTO)

[Date of requesting appeal against examiner's decision of rejection] 01.03.2001

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

日本特許庁

THIS PAGE BLANK (USPTO)

(11)特許出願公開番号

(43)公開日 平成9年(1997)1月21日

3 2 2 G

審査請求 未請求 請求項の数6 OL (全 16 頁)

(74) 代理人 弁理士 小川 勝男

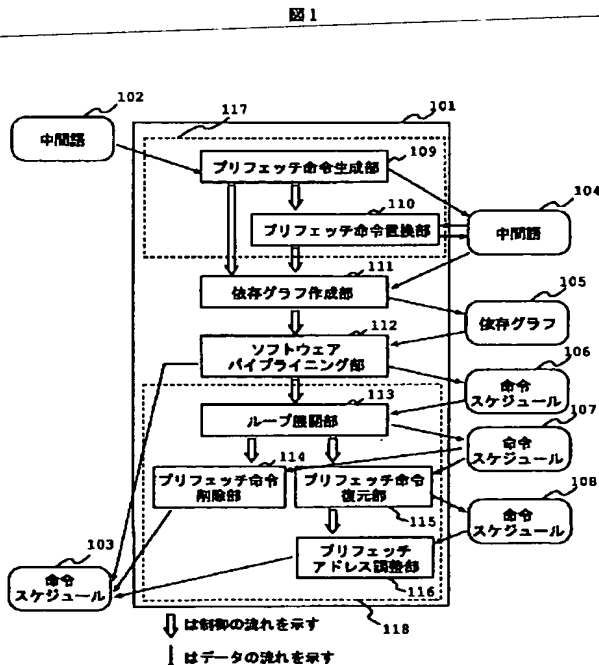
(54)【発明の名称】 データプリフェッチ方法

(57) 【要約】

【目的】—他の命令の実行と同時に主記憶からキャッシュメモリにデータを移動する機能を持つプリフェッチ命令を効率良くスケジュールする。

【構成】 プリフェッチ命令とメモリ参照命令の間に依存を持つ依存グラフを作成してソフトウェアパイプライニングを適用するか、あるいは、さらに、構成されたスケジュールのカーネル部を展開して冗長なプリフェッチ命令を削除するか、あるいは、複数のプリフェッチ命令をまとめて仮想的なプリフェッチ命令に置き換えて依存グラフを作成したものにソフトウェアパイプライニングを適用して構成されたスケジュールのカーネル部を展開した仮想的なプリフェッチ命令を元のプリフェッチ命令に置き換えてプリフェッチ命令をスケジュールする。

【効果】 キャッシュメモリ参照ミスと命令間の遅延による待ちを減少させ、計算機プログラムの実行の高速化に有効である。



【特許請求の範囲】

【請求項1】他の命令の実行と並行して主記憶からキャッシュメモリにデータを転送するプリフェッチ命令を有する計算機上で実行されるプログラムをコンパイルするコンパイラにおいて、

- (a)プログラム中のループのソースプログラムを中間コードへ変換し、
- (b)前記変換された中間コードに基づいて、プリフェッチ命令とメモリ参照を行なう命令との間に枝を持つ依存グラフを作成し、
- (c)ループの異なる繰返しを重ね合わせて実行することにより命令間の遅延が可能な限り隠蔽されるように命令をスケジューリングするソフトウェアパイプラインングを、プリフェッチ命令とメモリ参照を行なう命令の間の遅延をプリフェッチに要するサイクル以上にするとという制約を設けて、前記依存グラフに適用することにより、命令スケジューリングを行なうことを特徴とするデータプリフェッチ方法。

【請求項2】前記方法の前記ステップ(c)において、

- (c1)スケジューリングされたループ本体を複数回展開し、
- (c2)前記展開されたループ本体中の冗長なプリフェッチ命令を削除することを特徴とする請求項1記載のデータプリフェッチ方法。

【請求項3】他の命令の実行と並行して主記憶からキャッシュメモリにデータを転送するプリフェッチ命令を有する計算機上で実行されるプログラムをコンパイルするコンパイラにおいて、

- (a)プログラム中のループのソースプログラムを中間コードへ変換し、
- (b)プログラム中のループに含まれる複数のプリフェッチ命令を1つの仮想的なプリフェッチ命令に置き換え、
- (c)前記中間コードに基づいて、前記仮想的なプリフェッチ命令とメモリ参照を行なう命令との間に枝を持つ依存グラフを作成し、
- (d)ループの異なる繰返しを重ね合わせて実行することにより命令間の遅延が可能な限り隠蔽されるように命令をスケジューリングするソフトウェアパイプラインングを、前記依存グラフに適用して命令スケジューリングを行ない、
- (e)前記命令スケジューリングを展開することを特徴とするデータプリフェッチ方法。

【請求項4】前記方法の前記ステップ(e)において、

- (e1)得られたスケジューリングを複数回展開し、
- (e2)前記展開された仮想的なプリフェッチ命令を元の複数のプリフェッチ命令に置き換えることを特徴とする請求項3記載のデータプリフェッチ方法。

【請求項5】前記方法の前記ステップ(e)は、さらに、前記置き換えられたプリフェッチ命令が参照するアドレスを、前記プリフェッチ命令によるデータの転送が完了するのに十分先の繰返しで参照されるアドレスを参照

するように調節することを特徴とする請求項4記載のデータプリフェッチ方法。

【請求項6】他の命令の実行と並行して主記憶からキャッシュメモリにデータを転送するプリフェッチ命令を有する計算機上で実行されるプログラムをコンパイルするコンパイラにおいて、

前記プリフェッチ命令を含んだループをソフトウェアパイプラインングによってスケジューリングすることを特徴とするデータプリフェッチ方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明はデータプリフェッチ方法に関し、さらに詳しくは、ループに対してプリフェッチ命令をスケジューリングすることによりデータのプリフェッチを行ないプログラムの実行時間を短縮するために有効なコンパイル方法に関する。

【0002】

【従来の技術】計算機上でのプログラムの実行においては、命令間の依存関係によって生じる待ちと、メモリ参照によって生じる待ちが、実行時間の多くを占めている。

【0003】ループ内の命令間の依存関係による待ちは、ソフトウェアパイプラインングと呼ばれるスケジューリング手法によってかなり低減できる。ソフトウェアパイプラインングとは、例えば、ラマクリシュナン、ソフトウェアパイプラインング イン PA-RISC コンパイラ、ヒューレットパッカードジャーナル、39-45頁、1992年(S. Ramakrishnan, Software pipelining in PA-RISC compilers, Hewlett-Packard Journal, pp.39-45, 1992)に述べられているように、ループの異なる繰返しを重ね合わせて実行することにより命令間の依存関係による待ちを減少させ、命令実行の並列度を高める方式である。ソフトウェアパイプラインングが適用されたループでは、ループの実行開始前にプロローグと呼ばれる初期化処理を行なうコードを実行し、カーネルと呼ばれるコードを繰返すことによりループ本体の実行を行ない、ループの実行を終了した時点でエピローグと呼ばれるコードを実行して終了処理を行い、かつ、1回前の繰返しの完了を待たずに後続する繰返しの実行を開始することをその特徴とする。

【0004】命令間の依存関係による待ちに対して、メモリ参照による待ちはソフトウェアによる手法のみでは低減することが難しい。そのため、多くの計算機システムにおいては、メモリ参照による待ちを減少するために、キャッシュと呼ばれる高速少容量の記憶装置を主記憶とプロセッサの間に配置し、最近参照されたデータをキャッシュ上に置いて高速に参照できるようにしている。ただし、キャッシュを付加した場合においても、データの再利用がない場合などには、キャッシュミスが発生した場合の待ちは避けられない。

【0005】そこで、例えば、モウリー他、デザイン
アンド エバリュエーション オブ アコンパイラ アルゴ
リズム フォ プリフェッチング、プロシーディング オ
ブ ザ5 インターナショナル コンファレンス オン ア
ーキテクチュアル サポートフォ プログラミング ラン
ゲージ アンド オペレーティング システム、62-7
3頁、1992年(T.C.Mowry, et al., Design and Eva
luation of a Compiler Algorithm for Prefetching, Pr
ceedings of the 5th International Conference on A
rchitectural Support for Programming Languages and
Operating Systems, pp.62-73,1992)に述べられている
ように、主記憶からキャッシュにデータを先行的に移動
(プリフェッチ)する命令を利用することによって、メモ
リ参照による待ちを低減することが試みられている。

【0006】

【発明が解決しようとする課題】上記従来技術では、プ
リフェッチ命令をスケジューリングするための方式として、
プリフェッチ命令の遅延をループ本体の最短パスの長さ
で割った値以上の最小の整数回分だけ前の繰り返し前に
プリフェッチ命令が発行されるようにソフトウェアパイ
プラインングを適用すると述べられているが、実現の
詳細については明らかにされていない。

【0007】そこで、本発明の目的は、プリフェッチ命
令を含んだループにおけるプログラム実行時のメモリ参
照による待ちと命令間の依存関係による待ちを低減でき
る効率的な命令スケジューリング方法を提供することに
ある。

【0008】

【課題を解決するための手段】本発明の目的を達成する
ため、本発明では、プログラムのコンパイルの際に、以
下に示す3種の方式のいずれかにより、プログラム中の
ループに対してプリフェッチ命令のスケジューリングを
行なう。

【0009】キャッシュへのデータのプリフェッチはデ
ータの値に変更を加えない。このため、一般的なデータ
の定義と使用の関係に従うと、キャッシュへのデータの
プリフェッチと、ロード命令やストア命令によるメモリ
の参照との間には依存関係は存在しない。ただし、メモ
リ参照による待ちを隠蔽するには、プリフェッチ命令に
よるキャッシュへのデータの転送の完了後にメモリ参照
命令を発行しなければならないという制約から、プリフ
ェッチ命令とメモリ参照を行なう命令の間に、暗黙の依
存関係があるとみなすと既存のスケジューリング方式を
そのまま適用でき都合が良い。そこで、方式1では以下
のように、プリフェッチ命令とメモリ参照命令の間に依
存関係を設けてスケジューリングを行なう。

【0010】方式1:

(1)キャッシュミスを生じると予想されるメモリ参照命
令のそれぞれに対してプリフェッチ命令を生成する。

【0011】(2)上記(1)で生成したプリフェッチ命令

と、対応するメモリ参照命令の間に枝を持つ依存グラフ
を生成する。この際、プリフェッチ命令が発行されてか
ら、プリフェッチ命令によるキャッシュへのデータ転送
に要するサイクル数を経た後でメモリ参照命令が発行さ
れるよう、プリフェッチ命令とメモリ参照命令との間の
遅延を、プリフェッチ命令によるキャッシュへのデータ
転送に要するサイクル数以上に設定する。

【0012】(3)上記(2)で構成した依存グラフに対し
て、ソフトウェアパイプラインングを適用し、命令スケ
ジュールを得る。上記のように、ソフトウェアパイプ
ラインングとはループの異なる繰り返しを重ね合わせて実
行することにより、命令間の依存関係による待ちを減少
させる方式であり、これを上記(2)で構成した依存グラ
フへ適用することによって、プリフェッチ命令と対応す
るメモリ参照命令の間を十分離すことができる。

【0013】主記憶からキャッシュへのデータの転送単
位が一般的には32バイトや128バイトといった単位であ
るのに対して、ループ中での配列などの参照は4バイト
や8バイトといった、より小さな単位で行なわれること
が多い。このため、ループ中で配列などに対して連続的
なメモリ参照を行なう場合には、一度のプリフェッチ命
令によって、複数回分の繰り返しの実行で参照するデー
タを主記憶からキャッシュへ移動することができる場合
が多い。すなわち、一度のプリフェッチ命令によってN
回分の繰り返しの実行で参照するデータを主記憶からキ
ャッシュへ移動することができるとすると、プリフェッ
チ命令の発行はN回に1度行なえばよいことになる。

【0014】方式1で生成したスケジュールでは、繰り
返し1回毎にプリフェッチ命令を発行するため、冗長な
プリフェッチ命令が多数発行されることになってしま
う。そこで、方式2および方式3では、ループ本体を展
開し冗長なプリフェッチ命令がなるべく発行されないよ
うにプリフェッチ命令をスケジュールする。

【0015】まず、方式2では、上記方式1の(1)から
(3)までの処理によって生成されたソフトウェアパイプ
ライン化されたプリフェッチ命令を含むループのカーネ
ル部を展開し、冗長なプリフェッチ命令を取り除くこと
で無駄なプリフェッチ命令が発行されないようにする。

【0016】方式2:

(4)プリフェッチ命令の発行は、一回のプリフェッチ命
令でプリフェッチできるデータ数をNとすると、N回の
繰り返し毎に行なえばよいので、まず、上記(3)で構成
されたソフトウェアパイプライン化されたスケジュール
のカーネル部を展開し、その展開数がNの倍数となるよ
うにする。

【0017】(5)上記(4)の展開コードでは、カーネ
ル部がNの倍数回展開され、展開されたカーネル部の1回
の繰り返しで元のループのNの倍数回分の繰り返しが実
行されることになる。そこで、展開されたコードから、
N回に一回プリフェッチ命令が発行されるよう、冗長な

プリフェッチ命令を削除すれば、無駄なプリフェッチ命令の発行がなくなる。

【0018】方式2では、プリフェッチ命令にソフトウェアパイプラインングを適用した後で無駄なプリフェッチ命令を削除するので、プリフェッチ命令を削除したことによって命令間の距離がソフトウェアパイプラインングを適用した際に期待したものよりも短くなり、これによって命令間の依存などによる待ちが生じやすくなってしまう可能性がある。

【0019】そこで、方式3では、ループをソフトウェアパイプライン化してスケジュールを得た後でカーネル部を展開することを考慮して、まず、複数のプリフェッチ命令をまとめて1つの仮想的なプリフェッチ命令に置き換え、この仮想的なプリフェッチ命令を含んだ依存グラフを作成する。ただし、方式1や方式2の場合と異なり、方式3では仮想的なプリフェッチ命令と、対応するメモリ参照命令との間に依存は設けなくともよい。

【0020】次に、ソフトウェアパイプラインングを依存グラフに適用し、ソフトウェアパイプライン化されたスケジュールを得、カーネル部の展開数が一度のプリフェッチ命令でプリフェッチできるデータ数の倍数となるように必要に応じてループを展開する。展開された仮想的なプリフェッチ命令を元のプリフェッチ命令に置き換え、プリフェッチ命令が、対応するメモリ参照命令よりも十分前の繰り返しで発行されるよう、プリフェッチ命令が参照するアドレスを調節する。

【0021】これにより、方式2で命令を削除したことによって発生した命令間の依存を減少させる。

【0022】方式3:

(1) キャッシュミスを生じると予想されるメモリ参照命令それぞれに対してプリフェッチ命令を生成する。

【0023】(2) 上記(1)で生成したプリフェッチ命令を複数個組にして仮想的なプリフェッチ命令に置き換える。

【0024】(3) 元のループボディの命令と上記(2)で生成した仮想的なプリフェッチ命令からなる依存グラフを作成して、ソフトウェアパイプラインングを適用する。依存グラフの作成を行なう場合には、仮想的なプリフェッチ命令と、対応するメモリ参照命令との間の依存は考えなくともよい。

【0025】(4) 上記(3)で構成されたカーネル部の展開数が、一度のプリフェッチ命令でデータをプリフェッチできる繰り返し数の倍数となるように必要に応じてループの展開を行なう。展開後のスケジュールにおいて、仮想的なプリフェッチ命令は元のプリフェッチ命令を挿入する候補となる命令スロットを表している。

【0026】(5) 上記(4)の展開コード中にスケジュールされた仮想的なプリフェッチ命令を、元のプリフェッチ命令に置き換える。この置き換えは、一度のプリフェッチ命令でデータをプリフェッチできる繰り返し数の倍

数毎に、同一のプリフェッチ命令が発行されるようにする。これによって無駄なプリフェッチ命令の発行が抑制される。

【0027】(6) 上記(5)で置き換えたプリフェッチ命令が参照するアドレスをプリフェッチ命令によるデータの転送が完了するのに十分先の繰り返しで参照されるデータのアドレスとする。

【0028】

【作用】本発明の方法によれば、メモリの参照が連続的でない場合には、方式1によってプリフェッチ命令と対応するメモリ参照命令の間を十分離してソフトウェアパイプラインングを適用することができる。また、メモリの参照が連続的である場合には、方式2によってソフトウェアパイプラインングを適用した後で命令の削除を行なうか、方式3によって複数のプリフェッチ命令を仮想的なプリフェッチ命令に置き換えてソフトウェアパイプラインングを適用し、その後、元のプリフェッチ命令に復元することで無駄なプリフェッチ命令を発行を抑制し効率的にスケジュールすることができる。これにより本発明の目的を達成できる。

【0029】

【実施例】以下、図面を参照しながら本発明の一実施例について説明する。

【0030】図2は本発明の方法を実施する計算機の1つの例である。この例では、CPU201上で動作するコンパイラが外部記憶装置202からソースコード203を読み込み、これをオブジェクトコード204へ変換し外部記憶装置202へ格納する。

【0031】図3は、本発明によるデータのプリフェッチ方法を適用する計算機の1つの例である。CPU301で通常のメモリ参照命令の実行を行なう場合には、まず、キャッシュ302に参照対象のデータがあるかどうかを調べ、キャッシュ302にデータが存在すればそのデータを参照し、キャッシュ302に参照対象のデータが存在しなければ主記憶303上の当該データを参照すると共に、当該データの属するキャッシュブロックをキャッシュ302に置く。キャッシュの参照は主記憶の参照に比べて高速であり、参照対象のデータがキャッシュにあればメモリ参照によって発生する待ちを低減できる。

【0032】プリフェッチ命令は、他の命令の実行と同時に主記憶303からキャッシュ302へ参照対象のデータが属するキャッシュブロックを移動する命令である。主記憶303からキャッシュ302へキャッシュブロックを移動するのに十分なサイクル数だけ前にプリフェッチ命令を発行しておけば、主記憶303からキャッシュ302へのデータの転送を行なっている間に他の命令を実行できるので、当該データを参照するための待ちは無くなる。

【0033】図1に本発明の一実施例の構成を示す。図1において、スケジューリング処理部101はループ本体に対する中間語102を入力し、プリフェッチ命令を含ん

だメモリ間の依存およびメモリ参照による待ちの少ない命令スケジュール103を出力する。処理117および118は本発明に特徴的な処理であり、処理117ではプリフェッチ命令の生成と、スケジューリングの前処理を行ない、処理118では無駄なプリフェッチ命令の除去やプリフェッチアドレスの調整などの後処理を行なう。

【0034】始めに、方式1によりループをスケジュールする場合の実施例を示す。図4は方式1によってプリフェッチ命令のスケジュールを行なう場合の命令スケジューラの構成図である。方式1では、プリフェッチ命令生成部109が中間語102を入力し、ループ本体の中間語に含まれるメモリ参照命令のうち、キャッシュミスを起こす可能性の高いものについてプリフェッチ命令を生成して、プリフェッチ命令を加えた中間語104を構成する。

【0035】ここで、あるメモリ参照命令に対してキャッシュミスが起きる可能性があるかどうかについては、例えば、モウリ他、デザイン アンド エバリュエーションオブ ア コンパイラ アルゴリズム フォ プリフェッチング、プロシーディングオブ ザ 5 インターナショナル コンファレンス オン アーキテクチャ サポート フォ プログラミング ランゲージ アンド オペレーティング システム、62-73頁、1992年(T.C. Mowry他、Design and Evaluation of a Compiler Algorithm for Prefetching, Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.62-73, 1992)で述べられている公知技術やプログラムの実行のトレースを用いて推測することができる。生成するプリフェッチ命令がプリフェッチするアドレスは、対応するメモリ参照命令と同じとする。

【0036】すなわち、ループ中のロード命令、

LOAD X[i]

がキャッシュミスを起こしやすいとすると、同一要素をプリフェッチする命令、

FETCH X[i]

を作り、これを中間語に加える。

【0037】次に、依存グラフ作成部111では、プリフェッチ命令を含んだ中間語104を入力として依存グラフ105を作成する。この際、プリフェッチ命令と対応するメモリ参照命令との間に必要な遅延が主記憶からキャッシュへキャッシュブロックを転送するのに要する時間以上であることを表す枝を、プリフェッチ命令と対応するメモリ参照命令との間に設ける。次に、依存グラフ105に対してソフトウェアパイプライン部112でソフトウェアパイプラインを適用して、ソフトウェアパイプライン化された命令スケジュール 103を得る。

【0038】以上のように、プリフェッチ命令と対応するメモリ参照命令の間に、必要な遅延が主記憶からキャッシュへキャッシュブロックを転送するのに要する時間以上であることを表す枝を設けた依存グラフを作成する

ことにより、ソフトウェアパイプラインを適用する際に、プリフェッチ命令とそれに対応するメモリ参照命令との間が、主記憶からキャッシュへキャッシュブロックを転送するのに要する時間だけ離れることが保証されるので、メモリ参照による待ちが隠蔽されるようプリフェッチ命令をスケジュールすることができる。

【0039】上記説明におけるプリフェッチ命令生成部109を、図7に示す動作フローチャートを参照しつつ説明する。まず、ステップ701では処理すべきメモリ参照命令が残っているかどうかを判定し、あればステップ702へ制御を移し、なければ処理を終了する。ステップ702では、処理すべきメモリ参照命令を選択して変数MIに記憶する。ステップ703では、MIに記憶されたメモリ参照命令がキャッシュミスを起こす可能性が高いかどうかを判定し、キャッシュミスを起こす可能性が高いならばステップ704へ制御を移し、低い場合にはステップ701へ制御を移して次のメモリ参照命令を処理する。ステップ704ではMIに記憶したメモリ参照命令と同じアドレスを参照するプリフェッチ命令を作成する。

【0040】次に、方式2によりループをスケジュールする場合の実施例を説明する。方式2では、方式1の処理によって得られたソフトウェアパイプライン化された命令スケジュール 106のカーネル部を、ループ展開部113で複数回展開して、命令スケジュール107を得る。このループ展開数は、1度のプリフェッチ命令の実行で主記憶からキャッシュへ移動可能なキャッシュブロックの大きさをB、メモリ参照命令によって参照される要素の大きさをD、配列参照の要素の増分値をNとすると、例えば、 B/D とNの最小公倍数とすればよい。

【0041】ループの展開を行なうと、続いて、プリフェッチ命令削除部114により、ループを展開して得られた命令スケジュール 107から冗長なプリフェッチ命令を削除する。これにより、冗長なプリフェッチ命令を含まない最終的な命令スケジュール103が得られる。この冗長なプリフェッチ命令の削除に関しては、プリフェッチ命令は B/D 回に1度発行すれば十分であることから、展開された各々のプリフェッチ命令に対して、 B/D 個おきにプリフェッチ命令が発行されるよう、それ以外の命令を削除する。

【0042】以上に述べた方式では、ループ展開数が多くなる場合もあり得るので、ループ展開数を低く押えたい場合には、例えば、適当な回数だけループを展開して、先に述べたように、 B/D 個回の繰り返しおきにプリフェッチ命令が発行されるように、それ以外の命令を削除すれば、多少冗長なプリフェッチ命令が発行されることとなるが、展開数が大きくなるのを防ぐことができる。

【0043】上記説明におけるプリフェッチ命令削除部114の動作を、図8に示すフローチャートを参照しつつ

説明する。まず、ステップ801ではキャッシュブロックの大きさを定数B、参照対象要素の大きさを定数Dとして設定する。ステップ802では、未処理プリフェッチ命令が残っているかどうかを判定し、あればステップ803へ制御を移し、なければ処理を終了する。ステップ803では、図1におけるループ展開部113によってコピーされた同一の未処理プリフェッチ命令を順に変数 PFi ($0 \leq i \leq n$)に記録する。ステップ804では、 $0 \leq i \leq n$ に対して、 $(1 \text{ を } B/D \text{ で割った余り} : (i \bmod (B/D)) \neq 0)$ ならば、すなわち i が B/D の整数倍でなければプリフェッチ命令 PFi を削除し、ステップ802へ制御を移して次のプリフェッチ命令を処理する。これにより、プリフェッチ命令が B/D 回の繰り返しおきに発行されることになる。

【0044】次に、方式3によりループをスケジューリングする場合の実施例を説明する。方式3によりループをスケジューリングする場合には、まず、方式1と同様にプリフェッチ命令生成部109で、入力となる中間語102から、キャッシュミスを起こす可能性の高いメモリ参照命令に対するプリフェッチ命令を生成し、プリフェッチ命令を加えた中間語104を得る。

【0045】次に、プリフェッチ命令置換部110において、プリフェッチ命令生成部109で生成した複数のプリフェッチ命令を組にして仮想的なプリフェッチ命令に置き換える。この置き換えは、1度のプリフェッチ命令の実行で主記憶からキャッシュへ移動可能なキャッシュブロックの大きさをB、メモリ参照命令によって参照される要素の大きさをD、中間語104に含まれるプリフェッチ命令の数をMとすると、例えば、 $M/(B/D)$ 以上の最小の整数個の仮想的なプリフェッチ命令を作成し、 B/D 個毎のプリフェッチ命令と1つの仮想的なプリフェッチ命令を対応させる。仮想的なプリフェッチ命令を作成すると、中間語104中の元のプリフェッチ命令を削除し、新たに生成した仮想的なプリフェッチ命令を加える。

【0046】次に、依存グラフ作成部111では、中間語104を入力として依存グラフ105を生成する。この場合には、方式1および2の場合と異なり、仮想的なプリフェッチ命令とメモリ参照命令の間には依存は設けない。続いて、依存グラフ105を入力として、ソフトウェアパイプライン部112でループに対してソフトウェアパイプラインを適用し、ソフトウェアパイプライン化された命令スケジューリング106を得る。方式3では方式1および2の場合と異なり、プリフェッチ命令と対応するメモリ参照命令との間に依存関係を設けないので、ソフトウェアパイプラインを適用する際の命令配置の自由度が高くなる。

【0047】次に、ソフトウェアパイプライン化された命令スケジューリング106をループ展開部113で複数回展開し、命令スケジューリング107を得る。この展開数は、方式

2の場合と同様に、1度のプリフェッチ命令の実行で主記憶からキャッシュへ移動可能なキャッシュブロックの大きさをB、メモリ参照命令によって参照される要素の大きさをD、配列参照の要素の増分値をNとすると、例えば、 B/D とNの最小公倍数とすればよい。ループ展開部113によるループ展開処理が終了すると、得られた命令スケジューリング107に含まれる仮想的なプリフェッチ命令を、プリフェッチ命令復元部115においてプリフェッチ命令置換部110で置き換えた対応するプリフェッチ命令に復元する。ある仮想的プリフェッチ命令VPに、 n 個のプリフェッチ命令 $PF1, PF2, \dots, PF_n$ が対応しており、仮想的プリフェッチ命令VPが、ループ展開部113によって m 個の仮想的プリフェッチ命令 $VP1, VP2, \dots, VP_m$ に展開されているものとする、この復元処理は、例えば、以下のように行なわれる。

【0048】 $n=B/D$ の場合、 $j=i \bmod (B/D)$ とすると、 VP_i を PF_j に置き換える。 $n < B/D$ の場合、 $j=i \bmod (B/D)$ とすると、 $1 \leq j \leq n$ ならば VP_i を PF_j に置き換え、 $n < j$ ならば、 VP_i を削除する。

【0049】これによって、元のプリフェッチ命令からなる命令スケジューリング108が得られる。

【0050】次に、プリフェッチアドレス調整部116によって、プリフェッチ命令によるデータの転送が完了するのに十分先の繰り返しでデータがプリフェッチされるよう、命令スケジューリング108のプリフェッチ命令の参照対象のアドレスを調節し、冗長なプリフェッチ命令を含まない命令スケジューリング103を得る。

【0051】このアドレスの調節は、例えば、配列Xに対して、

FETCH X[i]

というプリフェッチ命令があった場合、次のように行なわれる。

【0052】すなわち、スケジューリングされたループの1回あたりの実行に要するサイクル数をL、プリフェッチ命令によって主記憶からキャッシュへ対象データのキャッシュブロックを転送するのに要するサイクル数をMとすると、 $M/L+(B/D)$ より以上の最小の整数回先繰り返しで参照する配列要素をプリフェッチするようにすれば良い。

【0053】すなわち、この繰り返し数を α とすると、上記のプリフェッチ命令の参照アドレスを、

FETCH X[i+ α]

のように調節すれば良い。

【0054】以下、上記方式3におけるプリフェッチ命令置換部110とプリフェッチ命令復元部115の処理について、フローチャートを参照しつつ説明する。

【0055】図9は、図1におけるプリフェッチ命令置換部110の動作フローチャートである。まず、ステップ901では、キャッシュブロックの大きさを定数B、参照対象要素の大きさを定数D、プリフェッチ命令の数を記録

するための変数 n の値を 0 として設定する。ステップ 902 では、プリフェッチ命令が残っているかどうか判定し、残っていればステップ 903 へ制御を移し、残っていなければ処理を終了する。ステップ 903 では、変数 n の値が 0 かどうか判定し、真であればステップ 904 へ制御を移し、偽であればステップ 906 へ制御を移す。ステップ 904 では、新たに仮想的なプリフェッチ命令を生成し、これを変数 VPF に記憶する。ステップ 905 では変数 VPF に記憶された仮想的プリフェッチ命令を中間語列に挿入する。

【0056】ステップ 906 ではプリフェッチ命令を選択し、変数 PF に記憶する。ステップ 907 では変数 PF に記録したプリフェッチ命令と変数 VPF に記録した仮想的なプリフェッチ命令を対応させる。ステップ 908 では中間語列から変数 PF に記録したプリフェッチ命令を削除する。ステップ 909 では変数 n の値を 1 だけ増す。ステップ 910 では、 n の値が B/D に等しいかどうか判定し、真であればステップ 911 へ制御を移し、偽であればステップ 902 へ制御を移して次のプリフェッチ命令を処理する。ステップ 911 では、 n の値を 0 に設定し、ステップ 902 へ制御を移して次のプリフェッチ命令を処理する。これによって、プリフェッチ命令が B/D 個に 1 個の割合で仮想的プリフェッチ命令に置き換えられる。

【0057】図 10 は、図 1 におけるプリフェッチ命令復元部 115 の動作フローチャートである。まず、ステップ 1001 ではキャッシュブロックの大きさを定数 B 、参照対象要素の大きさを定数 D として設定する。ステップ 1002 では仮想的プリフェッチ命令が残っているかどうか判定し、残っていればステップ 1003 へ制御を移し、残っていなければ処理を終了する。ステップ 1003 では、図 1 におけるループ展開部 113 によってコピーされた同一の仮想的プリフェッチ命令を順に変数 VP_i ($0 \leq i < m$) に記憶する。ステップ 1004 では、 VP_i に対応する元のプリフェッチ命令を変数 PF_j ($0 \leq j < n$) に記憶する。

【0058】ステップ 1005 では、プリフェッチ命令 PF_j の数 n が B/D であるかどうかを判定し、真であればステップ 1006 へ、偽であればステップ 1007 へ制御を移す。ステップ 1006 では、各 VP_i に対して、 $j = i \bmod (B/D)$ とすると、 VP_i を PF_j に置き換え、制御をステップ 1002 へ移して次の仮想的プリフェッチ命令を処理する。ステップ 1007 では、各 VP_i に対して、 $j = i \bmod (B/D)$ とすると、 $0 \leq j < n$ ならば VP_i を PF_j に置き換え、 $n \leq j$ ならば VP_i を削除して、制御をステップ 1002 へ移して次の仮想的プリフェッチ命令を処理する。これによって、仮想的なプリフェッチ命令が元のプリフェッチ命令に復元されると共に、各プリフェッチ命令が B/D 回の繰り返しが 1 度実行されるようになる。

【0059】図 11 は、図 1 におけるプリフェッチアドレス調整部 116 の動作フローチャートである。まず、ステップ 1101 ではキャッシュブロックの大きさを定数 B 、

参照対象要素の大きさを定数 D 、ループ 1 回あたりの実行サイクルを L 、主記憶からキャッシュへデータを転送するのに要するサイクル数を M 、プリフェッチ命令を先行して発行すべき繰り返し数 α を $M/L + (B/D)$ 以上の最小の整数として設定する。ステップ 1102 では、未処理プリフェッチ命令が残っているかを判定し、残っていればステップ 1103 へ制御を移し、残っていなければ処理を終了する。ステップ 1103 では、未処理のプリフェッチ命令を選択し、変数 PF へ記録する。ステップ 1104 では、変数 PF へ記憶したプリフェッチ命令で参照するアドレスを α 回先の繰り返して参照するアドレスへ変更する。これによって、メモリ参照命令が発行されるよりも十分先にプリフェッチ命令が発行されることになり、メモリ参照による待ちを隠蔽することができる。

【0060】続いて、具体例を用いて各方式の一実施例によるスケジューリングの効果について説明する。図 12 は実施例の説明に用いる FORTRAN プログラムのループの例である。このプログラムのループ本体から、例えば図 13 に示すような中間語が構成される。以下では、この中間語を入力とした場合の各方式によるプリフェッチ命令のスケジューリングの例を示す。

【0061】図 13 の例では、命令 1301、1302、1303 でメモリ参照が行なわれるが、このうち命令 1301 と命令 1303 が参照するアドレスは同一であるので、配列 X および配列 Y それぞれについて 1 つずつプリフェッチ命令を作成する。なお、この例ではメモリ参照命令およびプリフェッチ命令と演算命令を並列に実行できるスーパースカラ型のプロセッサを仮定する。ただし、本発明はスーパースカラ型のプロセッサのみに適用可能というわけではなく、逐次型のプロセッサや超多長命令形式 (VLIW) のプロセッサについても適用可能である。なお、以下の例では、一度のプリフェッチで 4 回分の繰り返して使用するデータをキャッシュに転送可能であり、主記憶からキャッシュへのデータの転送には 50 サイクル必要であると仮定する。

【0062】方式 1 :

(1) プリフェッチ命令の生成

配列 X および配列 Y について、プリフェッチ命令を生成する。プリフェッチ命令を付加した中間語は図 14 に示すようになる。この図で、命令 1401 および命令 1402 はそれぞれ配列 X および配列 Y に対するプリフェッチ命令である。

【0063】(2) 依存グラフの生成

図 15 にプリフェッチ命令を加えた中間語に対する依存グラフを示す。この図で、ノードは命令を表し、ノード間のエッジは依存関係を表している。各エッジの右に付加している数字は命令間を離すべきサイクル数を表している。この図に示すように、配列 X に対するプリフェッチ命令 1501 と配列 X に対するロード命令 1503 との間、および、配列 Y に対するプリフェッチ命令 1502 と配列 Y の

ロード命令1504との間に主記憶からキャッシュへデータを転送するのに必要な50サイクルの遅延を持つ依存関係を設ける。

【0064】(3)ソフトウェアパイプライン化

図15の依存グラフに対して、ソフトウェアパイプライン化されたスケジュールは図16のようになる。図16に示すスケジュールは、ループの初期化を行なうプロログ部1601、ループの繰り返しを行なうカーネル部1602、ループの終了処理を行なうエピログ部1603からなり、図16の各エントリは各サイクルに対応する命令スロットを表している。プリフェッチ命令は命令スロット1604および1605に割り当てられており、ソフトウェアパイプライン化によって、対応するメモリ参照命令の10回前の繰り返しで実行されるようにスケジュールされている。以上のように、ソフトウェアパイプライン化によって各命令間の依存を満たすようなスケジュールが得られたので、メモリ参照に伴う待ちが除去されることになる。

【0065】方式2：上記方式1の実施例では、1回の繰り返しにつき2つのプリフェッチ命令が発行されることになる。プリフェッチ命令では4回分の繰り返しで利用するデータをプリフェッチすることが可能なので、毎回プリフェッチ命令を発行するのは無駄である。そこで、方式2では以下の処理を方式1の結果に適用することで、無駄なプリフェッチ命令の発行を抑制する。

【0066】(4)ループ展開

方式1の(3)で構成されたソフトウェアパイプライン化されたループのカーネル部を展開する。本実施例では、一度のプリフェッチで4回分の繰り返しで参照するデータをキャッシュへ転送可能であると仮定しているので、繰り返し4回に1回の割合でプリフェッチ命令を発行すれば良い。そこで、カーネルを4回展開すると、図17に示すようなスケジュールが得られる。図17に示すスケジュールは、プロログ部1701、展開されたカーネル部1702、及びエピログ部1703からなる。カーネル部1702において配列Xに対するプリフェッチ命令は、命令スロット1704、1706、1708、1710に展開され、配列Yに対するプリフェッチ命令は、命令スロット1705、1707、1709、1711に展開されている。

【0067】(5)冗長なプリフェッチの削除

ループ展開された図17の命令スケジュールに対して、プリフェッチ命令の発行が4回の繰り返し毎に行なわれるよう配列Xおよび配列Yに対する冗長なプリフェッチ命令を削除する。これによって無駄なプリフェッチ命令の発行が抑制され、図18に示すようなスケジュールが得られる。図18において、プロログ部1802では冗長なプリフェッチ命令1805、1806、1807、1808、1810、1811が削除され、配列Xに対しては命令スロット1804、配列Yに対しては命令スロット1809のそれぞれのプリフェッチ命令によって、4回分の繰り返しのデータが無駄な

くプリフェッチされるようになる。

【0068】方式3：方式3では、無駄なプリフェッチ命令を発行しないことを考慮して以下のようにプリフェッチ命令のスケジュールを行なう。

【0069】(1)プリフェッチ命令の生成

プリフェッチ命令の生成は方式1の場合と同様に行なう。

【0070】(2)プリフェッチ命令の置換と依存グラフの作成

上記(1)で生成した複数のプリフェッチ命令を命令をまとめて仮想的なプリフェッチ命令を生成し、依存グラフを構成する。結果の依存グラフを図19に示す。図19に示すように、配列Xに対するプリフェッチ命令1901と、配列Yに対するプリフェッチ命令1902をまとめて仮想的なプリフェッチ命令1903に置き換える。方式1や方式2の場合と異なり、方式3では仮想的なプリフェッチ命令と対応するメモリ参照命令の間に依存関係は設けない。

【0071】(3)ソフトウェアパイプライン化

仮想的なプリフェッチ命令を加えたループ本体にソフトウェアパイプライン化を適用する。この結果、例えば、図20に示すようなソフトウェアパイプライン化されたスケジュールが得られる。図20に示すスケジュールは、プロログ部2001、カーネル部2002、及びエピログ部2003からなる。カーネル部2002の命令発行スロット2004には仮想的なプリフェッチ命令がスケジュールされている。

【0072】(4)ループ展開

方式2の場合と同様に上記(3)で構成したソフトウェアパイプライン化されたスケジュールのカーネル部を4回分展開する。これにより、図21のようなスケジュールが得られる。図21に示すスケジュールは、プロログ部2101、展開されたカーネル部2102、及びエピログ部2103からなる。ループ展開によってカーネル部2103にスケジュールされた仮想的なプリフェッチ命令は命令発行スロット2104、2105、2106、2107に展開されている。

【0073】(5)プリフェッチ命令の復元

図21のカーネル部2102の命令スロット2104、2105、2106、2107に展開された仮想的なプリフェッチ命令を元のプリフェッチ命令に置き換える。この結果を図22に示す。図21の命令スロット2104、2105、2106、2107に展開された仮想的なプリフェッチ命令は配列Xおよび配列Yに対するプリフェッチ命令を置き換えたものである。各々の配列に対するプリフェッチ命令が4回の繰り返しに1回発行されるように、図22の命令スロット2204および命令スロット2206にプリフェッチ命令を挿入する。この場合、元のプリフェッチ命令数が一度のプリフェッチで主記憶からキャッシュへ転送可能なデータを参照する繰り返し数に満たないので、展開された仮想的なプリフェッチ命令に対応する命令スロット2205と2207

については空きスロットとする。

【0074】(6)プリフェッチアドレスの調節
プリフェッチ命令の発行とプリフェッチ命令によってキャッシュに転送されるデータを参照する命令の発行とが、主記憶からキャッシュへのキャッシュブロックの転送が終了するのに十分なサイクル数だけ離れて行なわれるようプリフェッチ対象のアドレスを調整する。ここで、主記憶からキャッシュへキャッシュブロックを転送するのに要するサイクルは50サイクルで、1回あたりの繰返しに要するサイクルは4サイクルであるので、図22に示すように、ここでは14回先の繰返しで参照されるデータをプリフェッチするようにプリフェッチ命令の参照先を変更している。

【0075】以上の説明により、図1の中間語102を入力としてスケジューラ101によりプリフェッチ命令を含んだ命令スケジューラ103を作成することができる。すなわち、ループ繰返しにおいて、データの参照が連続的でない場合には方式1を用いることで、プリフェッチ命令と対応するメモリ参照命令を、主記憶からキャッシュへデータを転送するのに必要なサイクル数だけ離して発行することができるので、メモリ参照に伴う待ちを隠蔽することができる。また、データの参照が連続的な場合には、方式2および3を用いることで、冗長なプリフェッチ命令の発行を抑制することができる。さらに、方式3では方式2と比較して、仮想的なプリフェッチ命令とメモリ参照命令との間に依存関係を設けないので、命令配置の自由度が高くなり、また、カーネル部の展開を考慮してソフトウェアパイプラインングを適用するので命令間の依存などによる待ちの発生を低く抑えることができる。

【0076】

【発明の効果】本発明によれば、プリフェッチ命令を効果的にスケジューラし、プログラムの実行時のメモリ参照等による待ちを減少することができる。これにより計算機プログラムの実行の高速化に効果がある。

【0077】すなわち、本発明の方法によれば、メモリの参照が連続的でない場合には、方式1によってプリフェッチ命令と対応するメモリ参照命令の間を十分離してソフトウェアパイプラインングを適用することができる。また、メモリの参照が連続的である場合には、方式2によってソフトウェアパイプラインングを適用した後で命令の削除を行なうか、方式3によって複数のプリフェッチ命令を仮想的なプリフェッチ命令に置き換えてソフトウェアパイプラインングを適用し、その後、元のプリフェッチ命令に復元することで無駄なプリフェッチ命令の発行を抑制し効率的にスケジューラすることができる。

【図面の簡単な説明】

【図1】プリフェッチ命令をスケジューラする命令スケジューラの構成図である。

【図2】本発明を実施する計算機システムの例である。

【図3】本発明の対象とする計算機システムの例である。

【図4】方式1による命令スケジューラの構成図である。

【図5】方式2による命令スケジューラの構成図である。

【図6】方式3による命令スケジューラの構成図である。

【図7】プリフェッチ命令生成部のフローチャートである。

【図8】プリフェッチ命令削除部のフローチャートである。

【図9】プリフェッチ命令置換部のフローチャートである。

【図10】プリフェッチ命令復元部のフローチャートである。

【図11】プリフェッチアドレス調節部のフローチャートである。

【図12】FORTRANソースプログラムの例である。

【図13】中間語の例である。

【図14】プリフェッチ命令を含んだ中間語の例である。

【図15】方式1の依存グラフの例である。

【図16】方式1のソフトウェアパイプライン化されたスケジューラの例である。

【図17】方式2の展開したスケジューラの例である。

【図18】方式2の冗長なプリフェッチ命令削除を行なったスケジューラの例である。

【図19】方式3の依存グラフの例である。

【図20】方式3のソフトウェアパイプライン化されたスケジューラの例である。

【図21】方式3の展開したスケジューラの例である。

【図22】方式3のプリフェッチ命令の置換えを行なったスケジューラの例である。

【符号の説明】

101：ループに対するスケジューリング処理部

109：プリフェッチ命令生成部

110：プリフェッチ命令置換部

111：依存グラフ作成部

112：ソフトウェアパイプライン部

113：ループ展開部

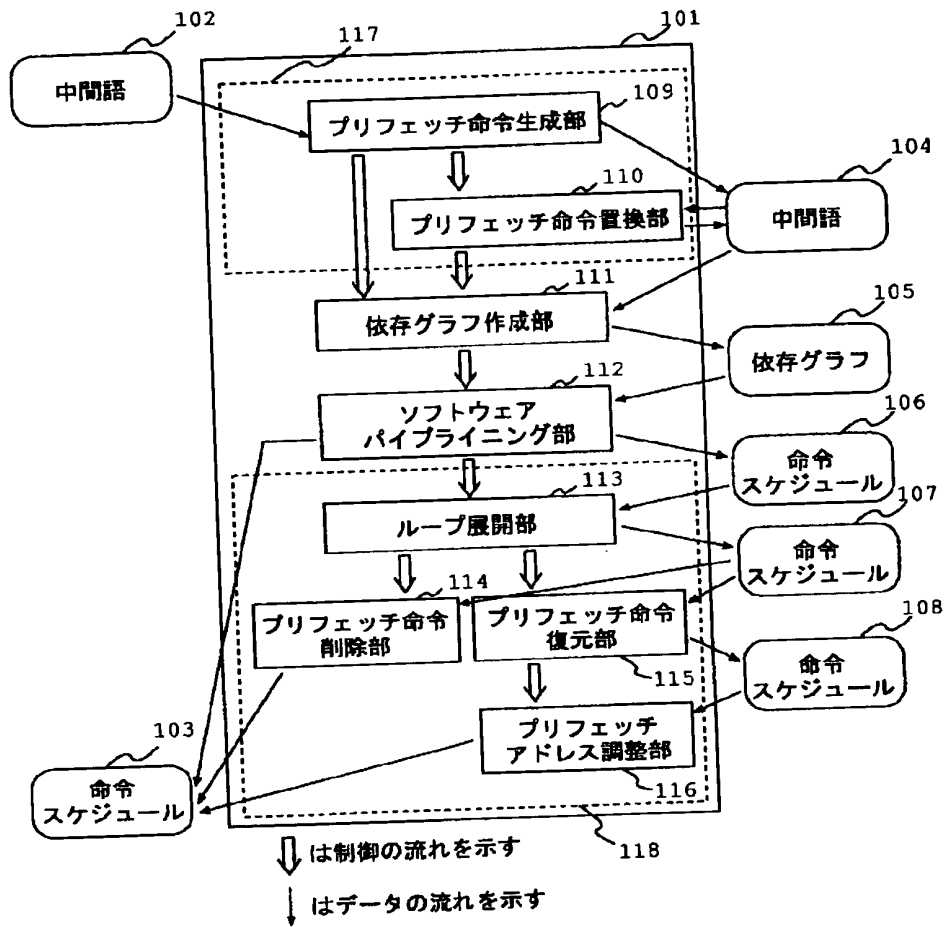
114：プリフェッチ命令削除部

115：プリフェッチ命令復元部

116：プリフェッチアドレス調整部

【図1】

図1



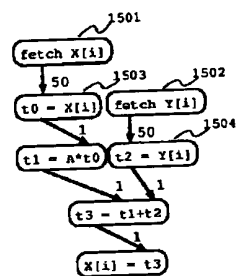
【図12】

図12

```
DO 10 I=0, N  
  X(I) = A*X(I)+Y(I)  
10 CONTINUE
```

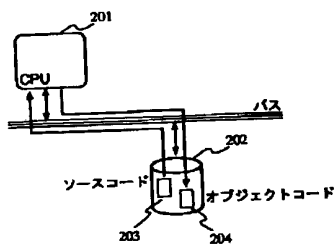
【図15】

図15



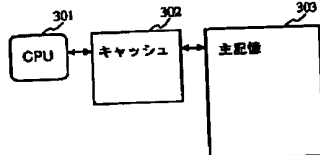
【図2】

図2



【図3】

図3



【図13】

図13

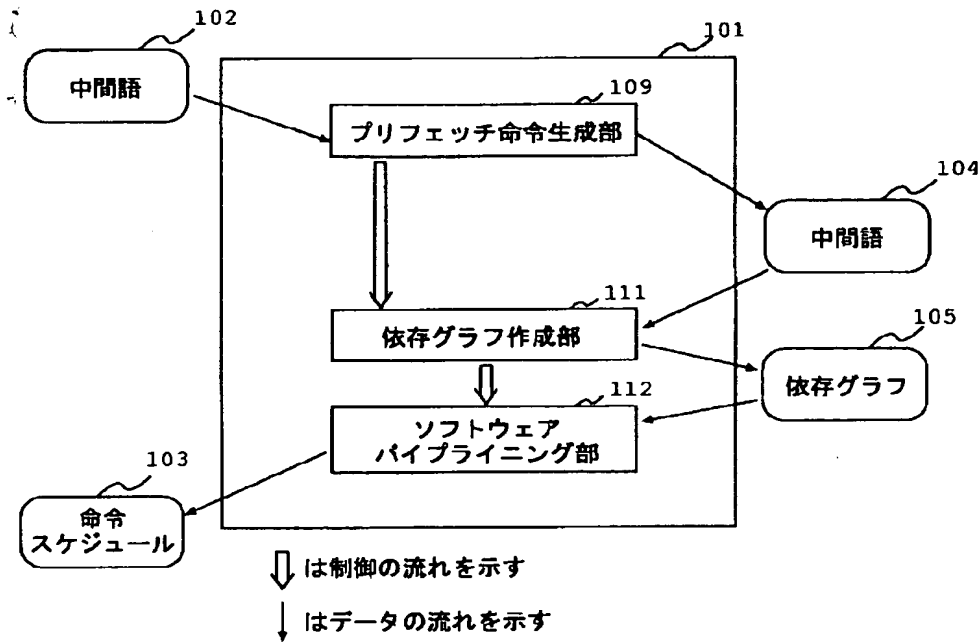
【図14】

図14

```
t0=X[i] ~ 1301  
t1=A*X[i]  
t2=Y[i] ~ 1302  
t3=t1+t2  
X[i]=t3 ~ 1303  
fetch X[i] ~ 1401  
fetch Y[i] ~ 1402  
t0=X[i]  
t1=A*X[i]  
t2=Y[i]  
t3=t1+t2  
X[i]=t3
```

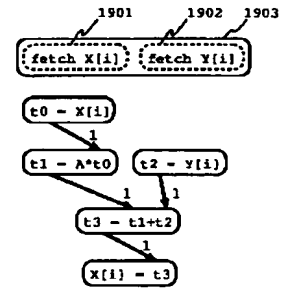
【図4】

図4



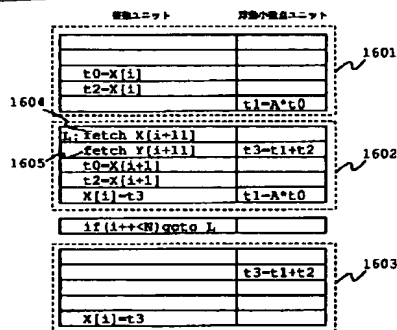
【図19】

図19



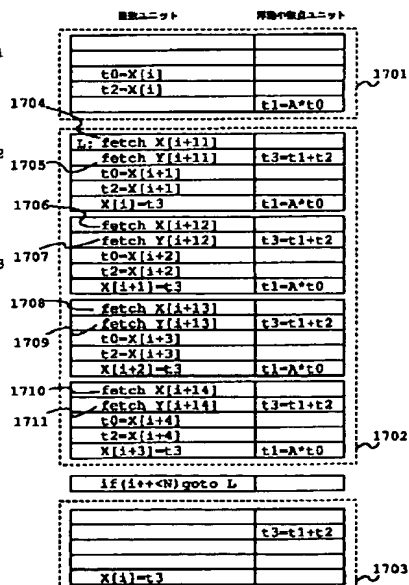
【図16】

図16



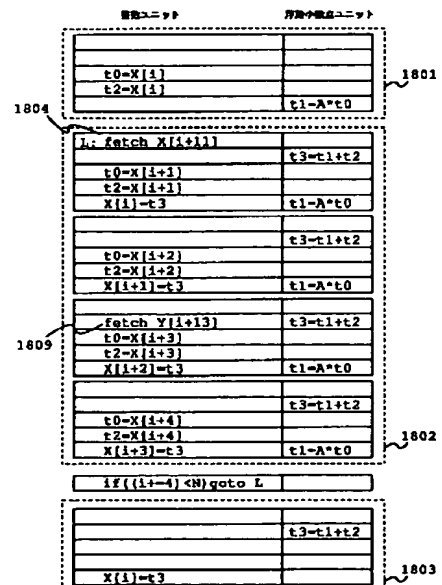
【図17】

図17



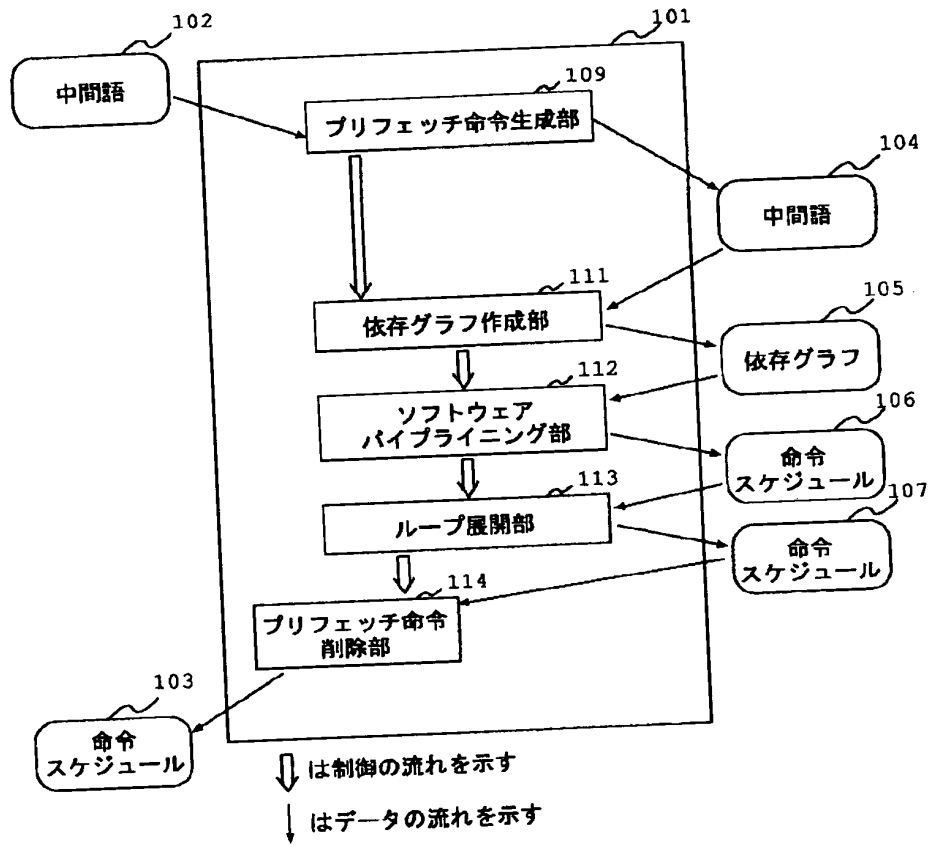
【図18】

図18



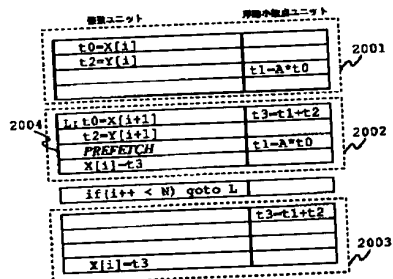
【図5】

図5



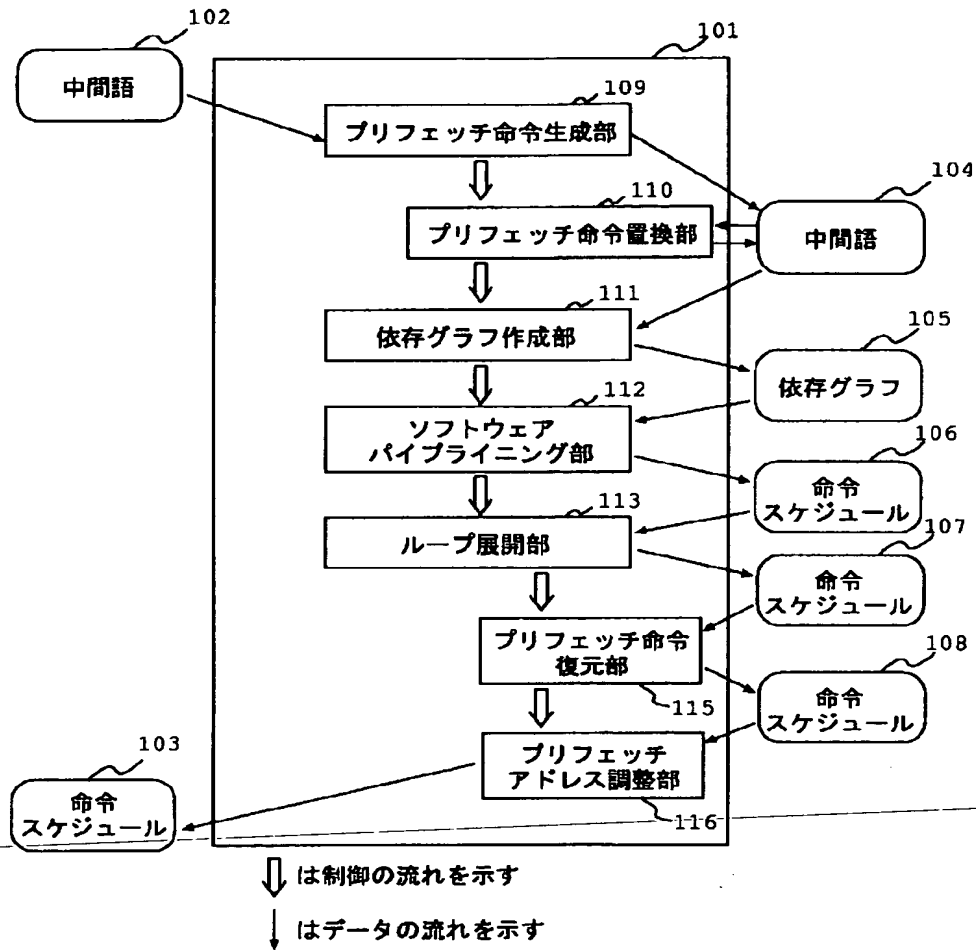
【図20】

図20



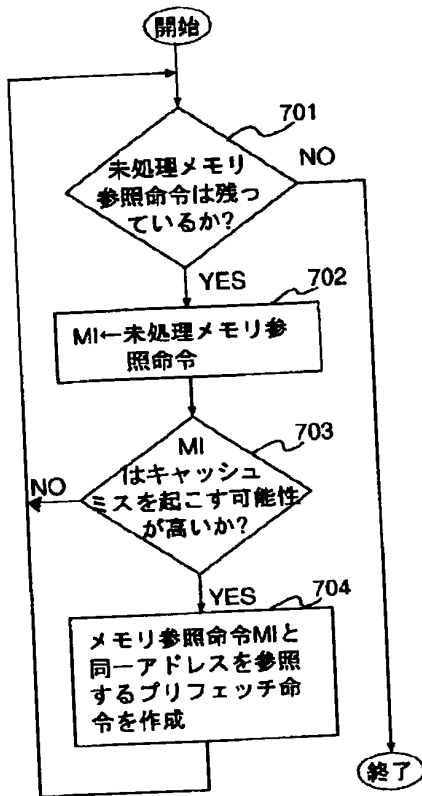
【図6】

図6



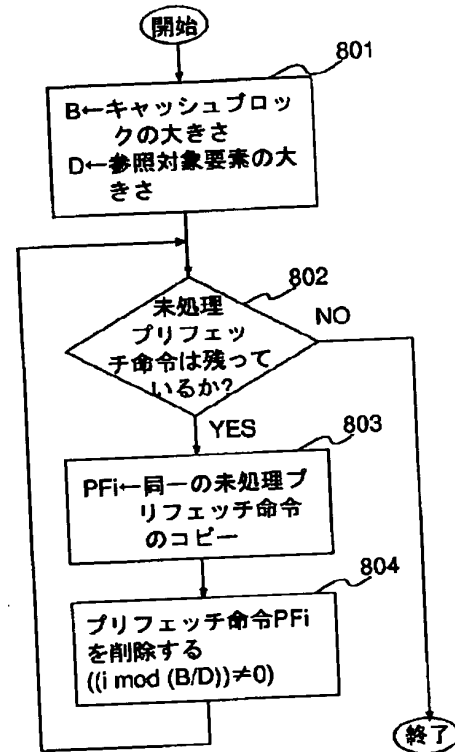
【図7】

図7



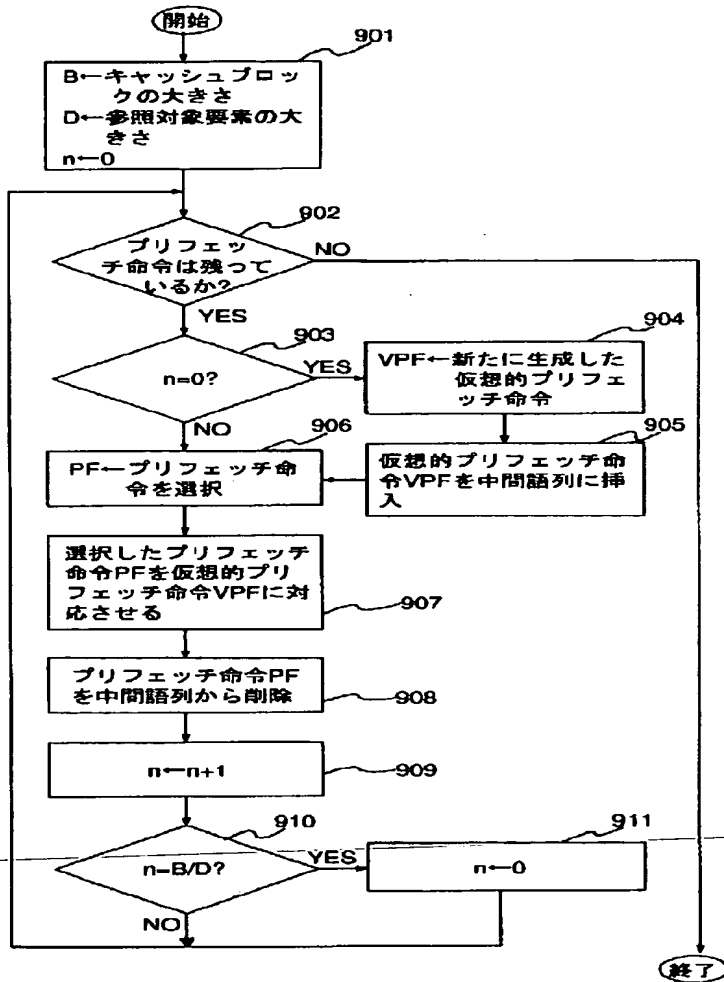
【図8】

図8



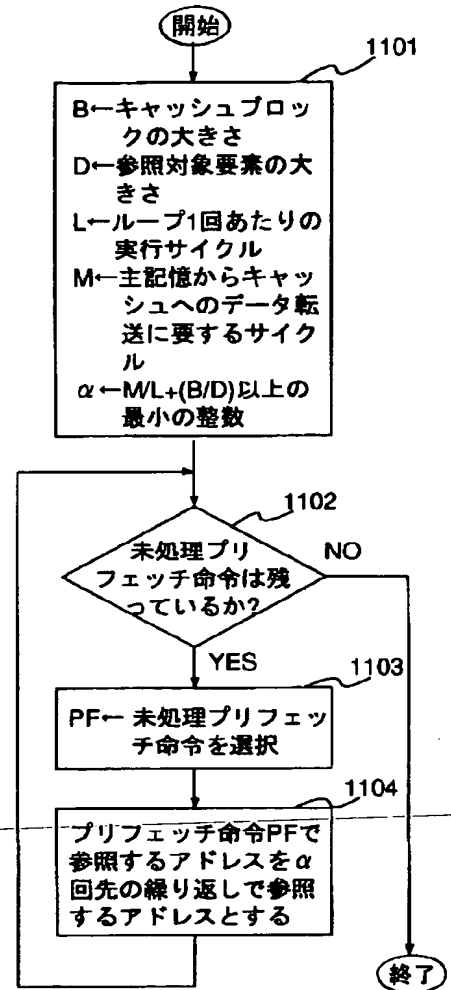
【図9】

図9



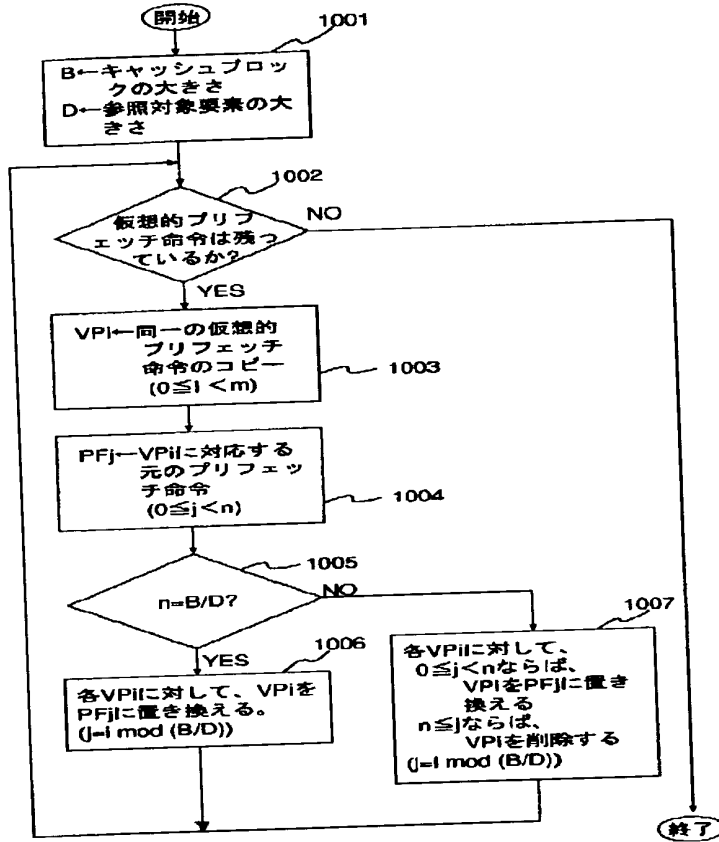
【図11】

図11



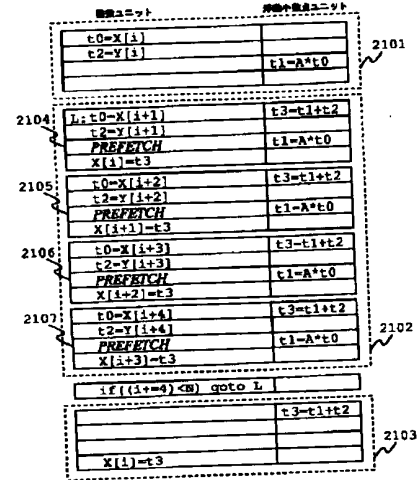
【図10】

図10



【図21】

図21



【図22】

図22

